PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 9/32

A1

(11) International Publication Number: **WO 99/40702**

(43) International Publication Date: 12 August 1999 (12.08.99)

(21) International Application Number: PCT/US99/02417

(22) International Filing Date: 4 February 1999 (04.02.99)

(30) Priority Data:
09/018,531 4 February 1998 (04.02.98) US

(71) Applicant: SUN MICROSYSTEMS, INC. [US/US]; 901 San Antonio Road, MS UPAL01-521, Palo Alto, CA 94303 (US).

(72) Inventors: HANNA, Stephen; 3 Beverly Road, Bedford, MA 01730 (US). PERLMAN, Radia; 10 Huckleberry Lane, Acton, MA 01720 (US).

(74) Agents: GARRETT, Arthur, S.; Finnegan, Henderson, Farrabow, Garrett & Dunner,, L.L.P., 1300 I Street, N.W., Washington, DC 20005-3315 (US) et al.

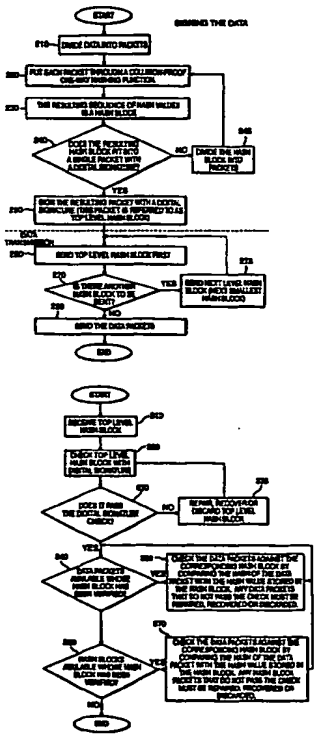
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: METHOD AND APPARATUS FOR EFFICIENT AUTHENTICATION AND INTEGRITY CHECKING USING HIERARCHICAL HASHING

(57) Abstract

In accordance with the present invention a method is provided for signing data and verifying data. Data signing involves dividing a data set into packets, hashing the data within each of the packets to produce a hash block including hash values and applying a digital signature to the hash block. The verifying process involves receiving a packet including data, a hash block and a digital signature, verifying the digital signature, hashing the data to produce hash values and comparing the hash values to values in the hash block.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

METHOD AND APPARATUS FOR EFFICIENT AUTHENTICATION AND INTEGRITY CHECKING USING HIERARCHICAL HASHING

BACKGROUND

A. Field of the Invention

This invention generally relates to data corruption detection and, more particularly, to a method and apparatus for efficient authentication and integrity checking in data processing using hierarchical hashing.

B. Description of the Related Art

Certain types of business activities create the need to transfer information in a secure manner. For instance, banks periodically "backup" their computer files to a remote central database and need to know that these files were successfully copied to the remote database without having been changed or corrupted during the process. Video conferencing is another example of an application that demands the secure transmission of information (video/voice/data).

Open networks such as the Internet provide simple and effective means for digital communication. However, such communication can be unintentionally corrupted by network transmission errors or altered by malicious acts. Several conventional techniques guard against these communication problems. These techniques require applying checksums or digital signatures to data before transmission and verifying the checksum or digital signature upon receipt.

Checksums, values derived from the data, are typically easy to compute. After computing the checksum, a transmitting machine sends the checksum with the data itself. A receiving machine then recalculates the checksum from the received data and compares the calculated checksum with the received checksum. However, a hacker with the ability to modify data likely also has the ability to replace the original checksum with a recalculated checksum that corresponds to the modified data.

Digital signatures protect against malicious acts intended to corrupt data but are more expensive than checksums to compute. The protection provided by digital signatures becomes increasingly important in networked environments where data must pass through unguarded points.

If the data is communicated in discrete packets signed with a bulk signature, detecting corruption or invalid packets inserted by a hacker also carries a high cost. It requires waiting for the receipt of at least one copy of each sequence number of packets. At this point, the signature can be checked. The integrity of the bulk signature cannot be checked until all of the packets are received. This aspect creates a time delay relative to the size of the data set. If several packets with the same sequence number and different contents are received, one is a forgery. Then, to detect which one is the valid copy of any given sequence number if a forgery is inserted, each copy of that sequence number would have to be compared with all the valid packets of the sequence.

A method called packet-level signature addresses this problem. The packet-level signature method assigns a digital signature to each individual packet. Although allowing the verification to begin as the individual packets are received making it easy to identify individual corrupted packets, this method requires additional computation and repeated checking of the digital signatures, which is quite time-consuming.

A conventional hierarchical hashing technique for neighboring databases on a local area network (LAN) allows a database management system to check if the databases are identical by hashing pieces of the database. The hashes are then hashed, and the final value is broadcast periodically to confirm that all neighbors on the LAN have identical databases. If they do not, the next hash level is compared until the area of the database that differs is located, at which time it can be updated accordingly. However, this system does not deal with the application of a single digital signature to protect an arbitrary amount of data against both malicious errors and unintentional errors. Currently, there is no system that allows a single digital signature to apply to an arbitrary amount of data and allows the data to be verified as it is received.

There is, therefore, a need for a system that protects against unintentional data corruption and malicious acts that cause errors in data processing and allows data to be checked as it is received. Additionally, the need exists for a system that provides information regarding which section of data was corrupted while keeping a low computational overhead.

SUMMARY OF THE INVENTION

Creating a hierarchy of hash values that start with packet hashes of an arbitrary data set and culminate in a single signed block allows a single digital signature to protect the data set from both data corruption and malicious acts that cause errors in data processing. Receiving this hierarchy of hashes before the data also allows the data packets to be quickly verified as they are received. The hierarchical structure used in the method cryptographically protects individual portions of the data and makes it easier to recognize corruption. Systems consistent with the present invention verify portions of the data even if other portions of the data are corrupt or have not yet been received.

In accordance with the present invention, as embodied and broadly described herein, a computer-implemented data processing method comprises the steps of dividing a data set into packets, hashing the data within each of the packets to produce a hash block including hash values and applying a signature to the hash block.

In accordance with another aspect of the present invention, as embodied and broadly described herein, a computer-implemented data processing method comprises the steps of receiving a packet including data, a hash block and a digital signature, verifying the digital signature, hashing the data to produce hash values and comparing the hash values to values in the hash block.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and, together with the description, serve to explain the advantages and principles of the invention. In the drawings,

FIG. 1 is a schematic block diagram illustrating a computer architecture suitable for use with the present invention;

FIG. 2 is a flowchart of the steps used to digitally sign the hash block in accordance with an implementation of the present invention;

DETAILED DESCRIPTION

Reference will now be made in detail to a system and method consistent with the present invention. Wherever possible, the same reference numbers will be used throughout the drawings and the following description to refer to the same or like parts.

Overview

Systems consistent with the present invention generally perform two data processing functions: (1) signing data and (2) verifying data. Such systems generally sign data by creating a hierarchical structure of hash blocks, sequences of hash values, which correspond to the given data. To accomplish this, a data set is divided into packets. A one-way, collision-proof hash function is applied to each packet which will result in a sequence of hash values. These hash values are grouped into packets, referred to as hash blocks. Higher level hash blocks are created by hashing the hash blocks. This process continues until a single hash block small enough to fit in a packet is achieved. The top level hash block, the smallest one, is signed with a digital signature. The hash blocks and data are then transmitted to an intended destination such as a network node.

Systems consistent with the present invention generally verify the data set by checking the digital signature on the top level hash block. Once the top level hash block is verified, the next lower level hash block can be verified by comparing the hash of the packets of that hash block against the hashes stored in the top level block. All lower level hash blocks are checked against the next higher level hash blocks in the same manner. Finally, the data is checked against the hash block containing the hashes of the data set. Any given data packet can be checked once all hash blocks above it are received.

Computer Architecture

Figure 1 is a block diagram that illustrates a computer system 100 upon which an embodiment of the invention may be implemented. Computer system 100 includes a bus 102 or other communication mechanism for communicating information, and a processor 104 coupled with bus 102 for processing information. Computer system 100 also includes a main memory 106, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 101 for storing information and instructions to be executed by processor 104. Main memory 106 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 104. Computer system 100 further includes a read only memory (ROM) 108 or other static storage device coupled to bus 102 for storing static information and instructions for processor 104. A storage device 110, such as a magnetic disk or optical disk, is provided and coupled to bus 102 for storing information and instructions.

Computer system 100 may be coupled via bus 102 to a display 112, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 114, including alphanumeric and other keys, is coupled to bus 102 for communicating information and command selections to processor 104. Another type of user input device is cursor control 116, such as a mouse, a trackball or cursor direction keys for communicating direction information and command selections to processor 104 and for controlling cursor movement on display 112. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

The invention is related to the use of computer system 100 for signing data and verifying data. According to one embodiment of the invention, signed or verified data is provided by computer system 100 in response to processor 104 executing one or more sequences of one or more instructions contained in main memory 106. Such instructions may be read into main memory 106 from another computer-readable medium, such as storage device 110. Execution of the sequences of instructions contained in main memory 106 causes processor 104 to perform the process steps described herein. In an alternative embodiment, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus

including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 110. Volatile media includes dynamic memory, such as main memory 106. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 102. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk. magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 104 for execution. For example, the instructions may initially be carried on magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 100 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector coupled to bus 102 can receive the data carried in the infra-red signal and place the data on bus 102. Bus 102 carries the data to main memory 106, from which processor 104 retrieves and executes the instructions. The instructions received by main memory 106 may optionally be stored on storage device 110 either before or after execution by processor 104.

Computer system 100 also includes a communication interface 118 coupled to bus 102. Communication interface 118 provides a two-way data communication coupling to a network link 120 that is connected to local network 122. For example, communication interface 118 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example,

communication interface 118 may be a local area network (LAN) card provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 118 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 120 typically provides data communication through one or more networks to other data devices. For example, network link 120 may provide a connection through local network 122 to a host computer 124 or to data equipment operated by an Internet Service Provider (ISP) 126. ISP 126 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 128. Local network 122 and Internet 128 both use electric, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 120 and through communication interface 118, which carry the digital data to and from computer system 100, are exemplary forms of carrier waves transporting the information.

Computer system 100 can send messages and receive data, including program code, through the network(s), network link 120 and communication interface 118. In the Internet example, a server 130 might transmit a requested code for an application program through Internet 128, ISP 126, local network 122 and communication interface 118. In accordance with the invention, one such downloaded application provides the data processing operations described herein. For example, data to be stored on computer-readable medium or transmitted to a remote device may be signed in accordance with the principles of the present invention. Additionally, data retrieved from a computer-readable medium or received from a remote device may be verified in accordance with the principles of the present invention.

The received code may be executed by processor 104 as it is received, and/or stored in storage device 110, or other non-volatile storage for later execution. In this manner, computer system 100 may obtain application code in the form of a carrier wave.

Hierarchical Hashing

Systems consistent with the present invention utilize a one-way, collision-proof hash function to transform data of arbitrary length into a fixed length hash value. One-way hash functions also have other characteristics. For example, it is relatively simple to apply the

messages to have the same hash value for the same function. One way and collision-resistant hash functions are described in detail in C. Kaufman, R. Perlman, M. Specine, "Network Security: Private Communication in a Public World," Prentice Hall, 1995.

(1) Signing Data

Fig. 2 is a flowchart of the steps used in the signing procedure of hierarchical hashing. Systems consistent with the present invention generally begin the hashing process by dividing the data set into small packets (step 210). The packet boundaries are generally arbitrary but, for more efficient use, should correspond to likely boundaries between good and corrupt data. Network datagram boundaries or disk sectors are examples of suitable boundaries.

Once the data is broken into packets, a collision-proof, one-way hash function is applied to each packet (step 220). Each application of the hash function will produce a single hash value. The application of the hash function to each of the data packets will produce a sequence of hash values. This sequence is called a hash block (step 230).

If the hash block created by the application of the hash function to the data packets is too large to fit in a single packet along with the digital signature (step 240), the hash block must be further broken down. The hash block itself is divided into packets (steps 245). The hash function is applied to each packet. Once again, the application of the hash function to the packets will produce a sequence of hash values. This new hash block will be smaller than the previous hash block and will be referred to as the next higher level hash block. If the next higher level hash block is too large to fit into a single packet with the digital signature, a new smaller hash block of a higher level is created by repeating steps 220-245.

The process of breaking down the hash block is repeated until the resulting hash block can fit in a single packet with the digital signature (step 240). This top level hash block 502 (see Fig. 5) is the smallest hash block. The hash block from which the top level hash block was created is referred to as the next level down hash block 501, with the complete structure of hash blocks forming a hierarchy of hash blocks 510.

If, however, the hash block originally created by the hashing of the data packets (step 230) is small enough to fit in a single packet with the digital signature, there is no need to go through the steps of breaking down the hash block and creating another higher level hash block and the top level hash block remains the only hash block.

Systems consistent with the present invention apply a digital signature to the top level hash block packet (step 250). No signature need be applied to any of the other hash blocks or data packets. The single digital signature applied to the top level packet is sufficient to verify all of the data. If extraordinary redundancy is desired, lower level hash blocks may be signed. These signatures would only be used if the top level hash block was corrupted and could not be recovered.

In one implementation for data transmission, the packets are transmitted in order beginning with the top level hash block packet (step 260), followed by each of the larger hash blocks (steps 270, 275, 280), and finally the data (step 280). For the fastest verification, however, the hash block packets and the data packets may be intermingled and need not be sent with all hash blocks first, but the top level packet should be sent first and lower level hashes be sent before the data they represent because their contents will need to be checked first, as explained below.

In accordance with the data signing procedure, the top level hash block packet with the single digital signature, the hierarchy of lower level hash blocks, and the data are sent to a receiver. This is all the information needed to protect the data.

(2) Verifying Data

Fig. 3 is a flowchart of the steps used in the data receiving and verification procedure for systems consistent with the present invention. To verify data, the digital signature on the top level packet must be verified first (step 320). If the signature fails this verification step or it is determined that the packet was corrupted during transmission, the top level packet must be repaired or recovered before checking the other packets (steps 330, 335). If lower level hash blocks were signed for extraordinary redundancy, their signatures may be checked in this case to allow verification to proceed.

If it is determined that additional hash blocks are expected (step 360), the packets of these hash blocks are then received and verified (step 370). The hash function is applied to each packet to derive its hash value. The hash value derived from the received hash block packet is compared with the one stored in the received top level hash block. If the comparison fails, the packet is corrupt and must be repaired or replaced before any packets that depend on this hash block can be verified (step 370).

If other hash blocks are expected, the hashes of the packets of each lower level hash block are compared with the values stored in the hash block in the level above. If this verification step fails, the packet is corrupt and must be repaired or recovered before any packets that depend on this packet can be verified. However, other packets in the hash block can be verified.

Examples

Reference will now be made to Figs. 4 and 5 to explain the procedures for hierarchical hashing and verification using examples. Figure 4 is a block diagram of example of the hashing of a sample set of data and signing of a hash block. Figure 5 is a diagram of a sample hierarchical structure of hash blocks and data based on the data in Fig. 4. In both Figs. 4 and 5, A_i (where$_i$=1,2,3...) represents data values while B_i and C_i represent hash values in different hash blocks. The packet size and amount of data in this sample is completely arbitrary. Thus, additional levels of hash blocks may be utilized depending on the size of the data and the selected hash function.

Given an initial data sequence of data values 401, the data values are divided into data packets 402a-f. The present example shows 18 data values, A_1 though A_{18}, divided into six packets with three data values in each packet. A one-way, collision-proof hash function 403 is applied to the data packets 403. The hashing of each data packet results in a single hash value. For instance, the hashing of the data packet A_1, A_2, A_3 in this case yields value B_1.

The hash values derived from the data packets, B_1 through B_6, form a hash block (404). In this example, the sequence of hash values in the hash block will not fit in a single packet with a digital signature so they are divided into two packets 404a and b of three hash values: B_1, B_2, and B_3 form one packet 404a and B_4, B_5, and B_6 form the second packet 404b.

A one-way, collision-proof hash function 405 is then applied to the hash block packets 405. Each hash block packet will form another hash value. For example, the packet B_1, B_2, B_3 will create the hash value C_1. These hash values, C_1 and C_2, form another hash block 406. This one, however, is small enough that it does not need to be broken down any further. Finally, the last hash block C_1, C_2, is signed with a digital signature 407 and results in the top level hash block with a digital signature 408. In this manner, the signing procedure (Fig. 2) is applied to data sequence 401 to generate top level hash block 408 and the next level hash block 404.

Data verification begins with checking the digital signature on the top level hash block 502. If the top level hash block 502 passes this check, the next level hash block 501 is verified. A hash function (not shown) is applied to each packet and compared with the corresponding hash value in the hash block in the level above it. For example, the packets 501a and b of the hash block 501, B_1 through B_6, are checked with the values stored in the other hash block 502, C_1 and C_2. In the sample in Fig. 5, the hash of the packet B_1, B_2, B_3 would be compared with the value C_1. If the check fails, the packet B_1, B_2, B_3 is corrupt and must be repaired or replaced before any packets that depend on it can be verified, in this case, data values A_1 through A_9, 500a-c.

The other hash block packet can be verified and remains unaffected by the corruption of the previous hash block packet. For instance, in Fig. 5, even if the first packet 501a in the hash block, B_1, B_2, B_3, was damaged, the second packet 501b in the same hash block, B_4, B_5, B_6, could still be checked by comparison of the hash of the packet 501b with the value C_2.

The data packets are checked in the same manner. For example, the data packet A_{10}, A_{11}, A_{12} would be checked by comparing the hash of the data packet 500d with the value B_4. If this check fails, the data packet A_{10}, A_{11}, A_{12}, is corrupt. However, the other data packets can still be verified. For an arbitrary example, in Fig. 5, even if the fourth packet 500d in the data, A_{10}, A_{11}, A_{12}, was corrupt, the fifth packet 500e in the data A_{13}, A_{14}, A_{15} could still be checked by comparison of the hash of the data packet with the value B_5. In this manner, the data receiving

received or retrieved out of order. Systems consistent with the present invention need not wait for all of the packets to be delivered to verify a data packet. Any packet can be verified as long as the hash block for that packet has been received and verified. This effectively reduces the delay time to the amount of time required to compute the hash and compare it to the value stored in its corresponding hash block. This delay time is much less than bulk signature's delay time in which all packets must be received before any can be verified. The computational overhead is less than packet-level signature.

The choice of when to send a hash block can save time. For instance, the second packet in a hash block need not be sent until after the packets that depend on the first packet in the hash block. This implementation can save the delay time of sending all of the hash blocks first.

Conclusion

Hierarchical hashing consistent with the present invention protects against malicious modification of data, while checksums do not. It also allows a single digital signature to apply to an arbitrary amount of data, which packet-level signature does not. The structure permits verification of data as it is received, thus eliminating the delay time of waiting for all of the data to be received as in bulk signature methods. Although the data can be verified as it is received, systems consistent with the present invention do not have the high computational overhead associated with individual packet signing. Additionally, the data need not be received or sent in any particular order for verification to begin.

The hierarchical structure allows for recognition of corrupt individual packets or sections of data. The other packets that are not corrupt can be used and are unaffected by the corrupt packets. Additionally, other packets that are unverified can be verified even though some packets may be corrupt. Corrupt packets can be repaired or recovered while other packets are being checked. Total replacement of the data is not needed as in bulk signature methods, thus creating greater efficiency and reducing time expended.

In summary, systems consistent with the present invention thus allow a single digital signature to protect an arbitrary amount of data, while cryptographically protecting individual portions of the data. To accomplish this, a hierarchy of hash values representing the data is built, and the top level of hashes are signed and sent. After receipt, the digital signature on the hash values is checked. The data, upon receipt, is checked against the hash values that corresponded to the data.

The foregoing description of an implementation of the invention has been presented for purposes of illustration and description. It is not exhaustive and does not limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practicing of the invention. The scope of the invention is defined by the claims and their equivalents.

2. The method of claim 1, wherein the applying step includes the substep of appending a digital signature to hash block.

3. The method of claim 1 further comprising the step of:

transmitting the data set and hash block.

4. The method of claim 1, wherein the hashing step includes the substep of determining whether the hash values fit into a single packet.

5. The method of claim 1, wherein the hashing step includes the substep of applying a collision-proof hash function.

6. The method of claim 1, wherein the hashing step includes the substeps of

(a) separating the hash values into packets; and

(b) applying a hash function to each of the packets.

7. The method of claim 6, further comprising the step of

repeating steps (a) and (b) until the resulting hash values fit into a single packet.

8. The method of claim 7, wherein the resulting sequence of hash values from each iteration of the steps (a) and (b) is smaller than the previous sequence of hash values.

9. The method of claim 8, wherein the packet size is large enough to hold at least two hash values.

10. A computer implemented data processing method comprising the steps of:

receiving a packet including data, a hash block and a digital signature;

verifying the digital signature;

hashing the data to produce hash values; and

comparing the hash values to values in the hash block.

11. A computer implemented data processing method comprising the steps of:

signing, with a digital signature, a hash block corresponding to a data set;

sending the data and the hash block from a source to a destination;

upon receipt at the destination, checking the digital signature on the hash block;

applying the hash function to the received data set; and

comparing the hashes of the received data with the hash values stored in the hash block.

12. The method of claim 11, wherein the signing step includes the substep of separating the data set into packets.

13. The method of claim 11, wherein the applying step includes the substep of utilizing a collision-proof hash function.

14. The method of claim 12, wherein the packet size is at least large enough to hold 2 hash values.

15. The method of claim 11, wherein the signing step includes the substeps of

(i) separating the hash block into packets; and

(ii) applying a hash function to the packets to create a second hash block.

16. The method of claim 15, wherein the steps (i) and (ii) are repeated until the resulting hash block can fit in a single packet with a digital signature.

17. A data processing apparatus comprising:

a dividing component configured to divide a data set into packets;

a hashing component configured to hash the data within each of the packets to produce a hash block including hash values; and

an applying component configured to apply a signature to the hash block.

18. The apparatus of claim 17, wherein the applying component includes:

an appending component configured to append a digital signature to hash block.

19. The apparatus of claim 17 further comprising:

a transmitting component configured to transmit the data set and hash block.

20. The apparatus of claim 17, wherein the hashing component includes:

a determining component configured to determine whether the hash values fit into a single packet.

(b) an applying component configured to apply a hash function to each of the packets.

23. The apparatus of claim 22, further comprising:

a performing component configured to perform the functions of components (a) and (b) repeatedly until the resulting hash values fit into a single packet.

24. A data processing apparatus comprising:

a receiving component configured to receive a packet including data, a hash block and a digital signature;

a verifying component configured to verify the digital signature;

a hashing component configured to hash the data to produce hash values; and

a comparing component configured to compare the hash values to values in the hash block.

25. A data processing apparatus comprising:

a signing component configured to sign, with a digital signature, a hash block corresponding to a data set;

a sending component configured to send the data and the hash block from a source to a destination;

a checking component configured to, upon receipt at the destination, check the digital signature on the hash block;

a applying component configured to apply the hash function to the received data set; and

a comparing component configured to compare the hashes of the received data with the hash values stored in the hash block.

26. The apparatus of claim 25, wherein the signing component includes:

a signing component configured to divide the data set into packets.

27. The apparatus of claim 25, wherein the applying component includes

a component configured to utilize a collision-proof hash function.

28. The apparatus of claim 26, wherein the packet size is at least large enough to hold 2 hash values.

29. The apparatus of claim 25, wherein the signing component includes:

(i) a separating component configured to separating the hash block into packets; and

(ii) an applying component configured to apply a hash function to the packets to create a second hash block.

30. The apparatus of claim 29, further including

a performing component configured to perform the functions of components (I) and (ii) repeatedly until the resulting hash block can fit in a single packet with a digital signature.

31. A computer product comprising a computer readable medium having computer readable code embodied therein for processing data, the computer readable medium comprising:

a dividing module configured to divide a data set into packets;

a hashing module configured to hash the data within each of the packets to produce a hash block including hash values; and

an applying module configured to apply a signature to the hash block.

32. The product of claim 31, wherein the applying module includes:

an appending module configured to append a digital signature to hash block.

33. The product of claim 31 further comprising:

a transmitting module configured to transmit the data set and hash block.

34. The product of claim 31, wherein the hashing module includes:

a determining module configured to determine whether the hash values fit into a single packet.

35. The product of claim 31, wherein the hashing module includes:

an applying module configured to apply a collision-proof hash function.

36. The product of claim 31, wherein the hashing module includes:

(a) a separating module configured to separate the hash values into packets; and

(b) an applying module configured to apply a hash function to each of the packets.

37. The product of claim 36, further comprising:

a performing module configured to perform the functions of modules (a) and (b) repeatedly until the resulting hash values fit into a single packet.

a comparing module configured to compare the hash values to values in the hash block.

39. A computer product comprising a computer readable medium having computer readable code embodied therein for processing data, the computer readable medium comprising:

a signing module configured to sign, with a digital signature, a hash block corresponding to a data set;

a sending module configured to send the data and the hash block from a source to a destination;

a checking module configured to, upon receipt at the destination, check the digital signature on the hash block;

an applying module configured to apply the hash function to the received data set; and

a comparing module configured to compare the hashes of the received data with the hash values stored in the hash block.

40. The product of claim 39, wherein the signing module includes:

a signing module configured to divide the data set into packets.

41. The product of claim 39, wherein the applying module includes

an applying module configured to utilize a collision-proof hash function.

42. The product of claim 40, wherein the packet size is at least large enough to hold 2 hash values.

43. The product of claim 39, wherein the signing module includes:

(i) a separating module configured to separate the hash block into packets; and

(ii) an applying module configured to apply a hash function to the packets to create a second hash block.

44. The product of claim 43, further including

a performing module configured to perform the functions of modules (I) and (ii) repeatedly until the resulting hash block can fit in a single packet with a digital signature.

45. A data structure stored in a computer readable medium for use by a processor in processing packaged data, the data structure comprising:

a plurality of packets, each including data;

a hash block including hash values resulting from application of a hash function to the packets of data; and

a digital signature appended to the hash block.

46. A data structure having portions capable of being stored separately in a computer readable medium for use by a processor in processing data, the data structure comprising:

a plurality of packets, each including data;

a first hash block including hash values resulting from application of a hash function to the packets of data, the first hash block being larger than a predetermined size based on the size of a single packet and divided into hash block packets; and

a second hash block including hash values resulting from application of a hash function to the packets of hash values of the first hash block, the size of the second hash block being within an acceptable range of the predetermined size.

47. The data structure of claim 46, further comprising:

a digital signature appended to the second hash block.

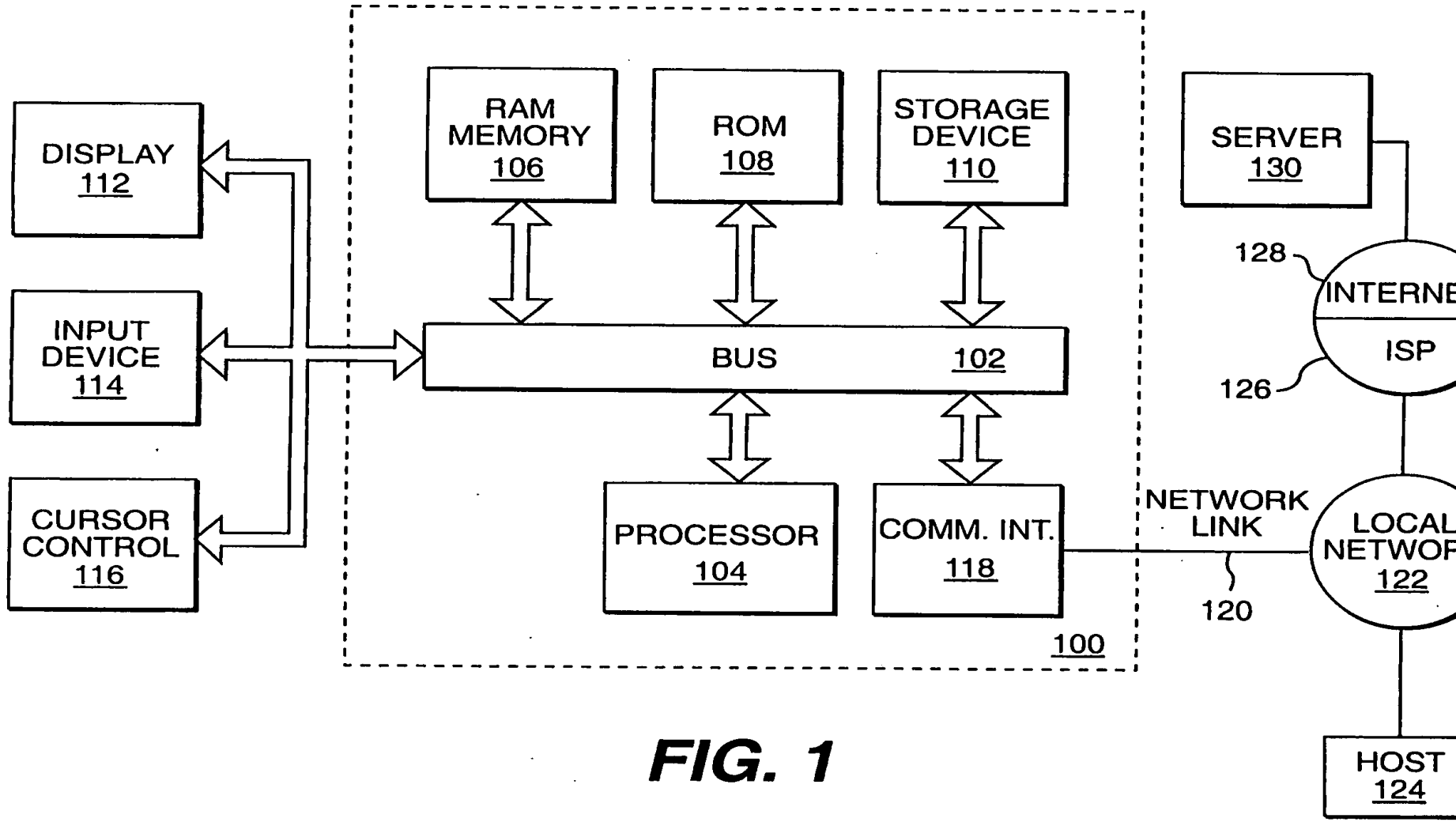


FIG. 1

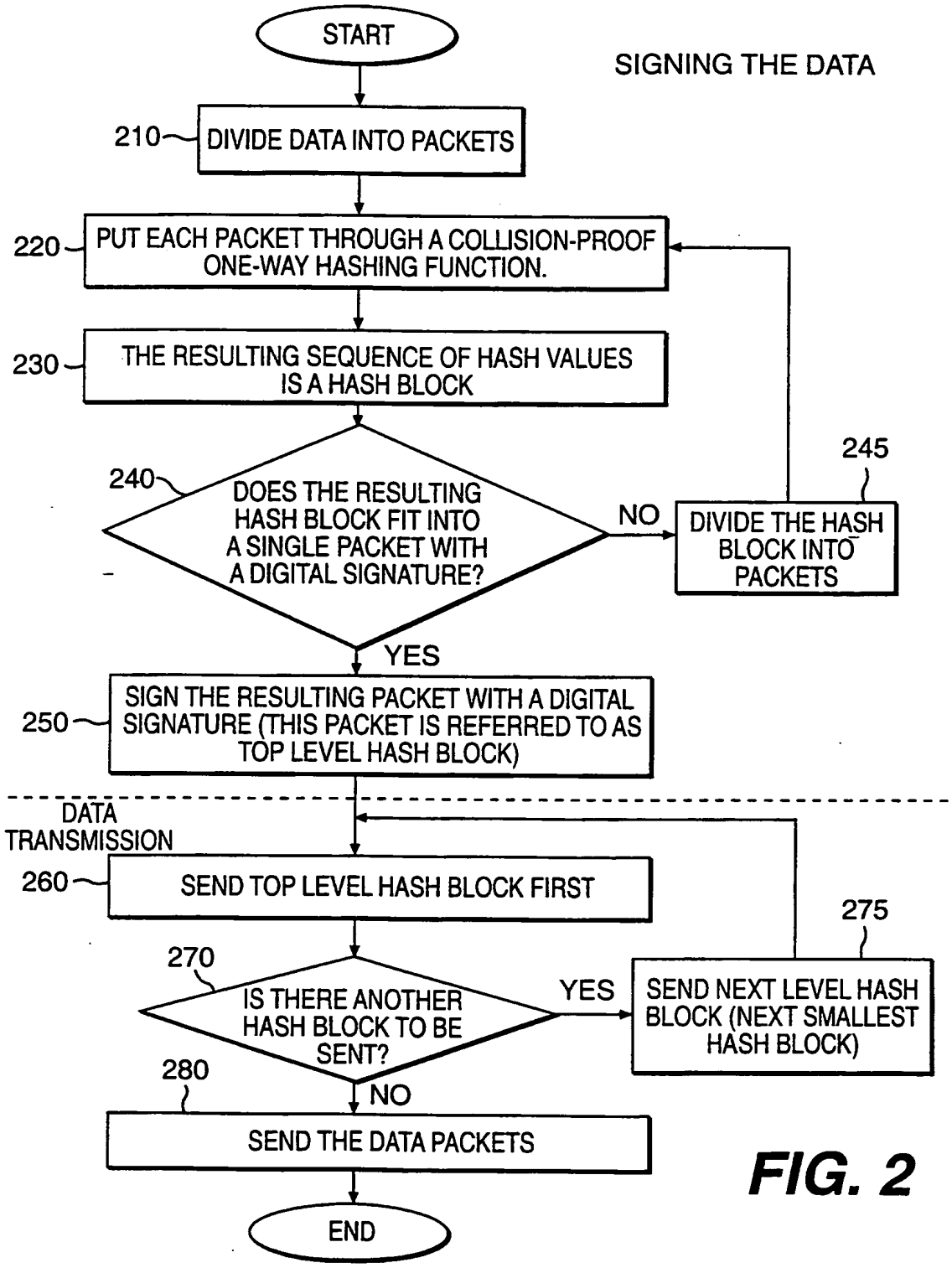

START
SIGNING THE DATA
210
DIVIDE DATA INTO PACKETS
220
PUT EACH PACKET THROUGH A COLLISION-PROOF ONE-WAY HASHING FUNCTION.
230
THE RESULTING SEQUENCE OF HASH VALUES IS A HASH BLOCK
240
DOES THE RESULTING HASH BLOCK FIT INTO A SINGLE PACKET WITH A DIGITAL SIGNATURE?
NO
245
DIVIDE THE HASH BLOCK INTO PACKETS
YES
250
SIGN THE RESULTING PACKET WITH A DIGITAL SIGNATURE (THIS PACKET IS REFERRED TO AS TOP LEVEL HASH BLOCK)
DATA TRANSMISSION
260
SEND TOP LEVEL HASH BLOCK FIRST
270
IS THERE ANOTHER HASH BLOCK TO BE SENT?
YES
275
SEND NEXT LEVEL HASH BLOCK (NEXT SMALLEST HASH BLOCK)
280
NO
SEND THE DATA PACKETS
END

FIG. 2

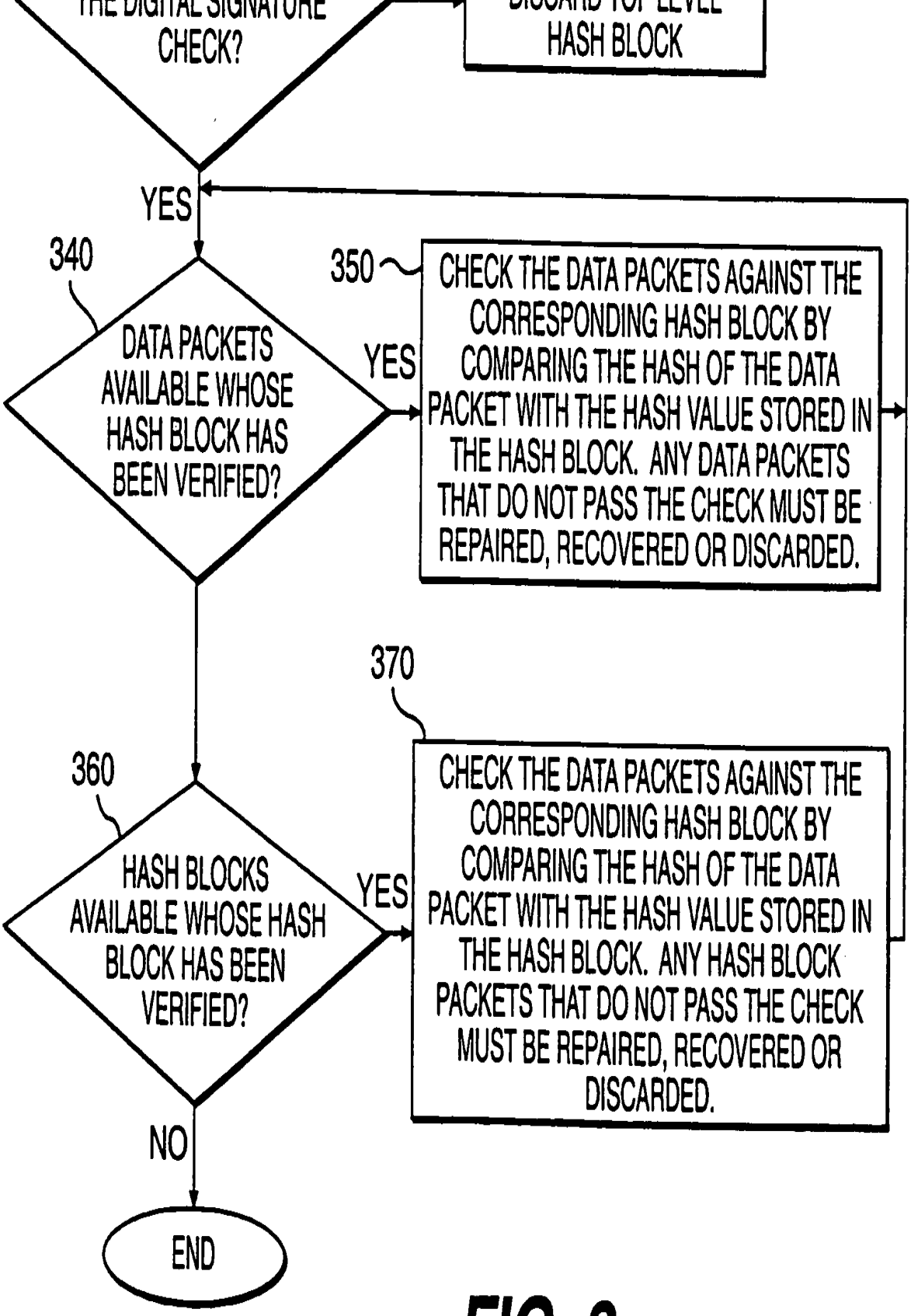

THE DIGITAL SIGNATURE CHECK?
HASH BLOCK
YES
340
DATA PACKETS AVAILABLE WHOSE HASH BLOCK HAS BEEN VERIFIED?
YES
350
CHECK THE DATA PACKETS AGAINST THE CORRESPONDING HASH BLOCK BY COMPARING THE HASH OF THE DATA PACKET WITH THE HASH VALUE STORED IN THE HASH BLOCK. ANY DATA PACKETS THAT DO NOT PASS THE CHECK MUST BE REPAIRED, RECOVERED OR DISCARDED.
370
360
HASH BLOCKS AVAILABLE WHOSE HASH BLOCK HAS BEEN VERIFIED?
YES
CHECK THE DATA PACKETS AGAINST THE CORRESPONDING HASH BLOCK BY COMPARING THE HASH OF THE DATA PACKET WITH THE HASH VALUE STORED IN THE HASH BLOCK. ANY HASH BLOCK PACKETS THAT DO NOT PASS THE CHECK MUST BE REPAIRED, RECOVERED OR DISCARDED.
NO
END

FIG. 3

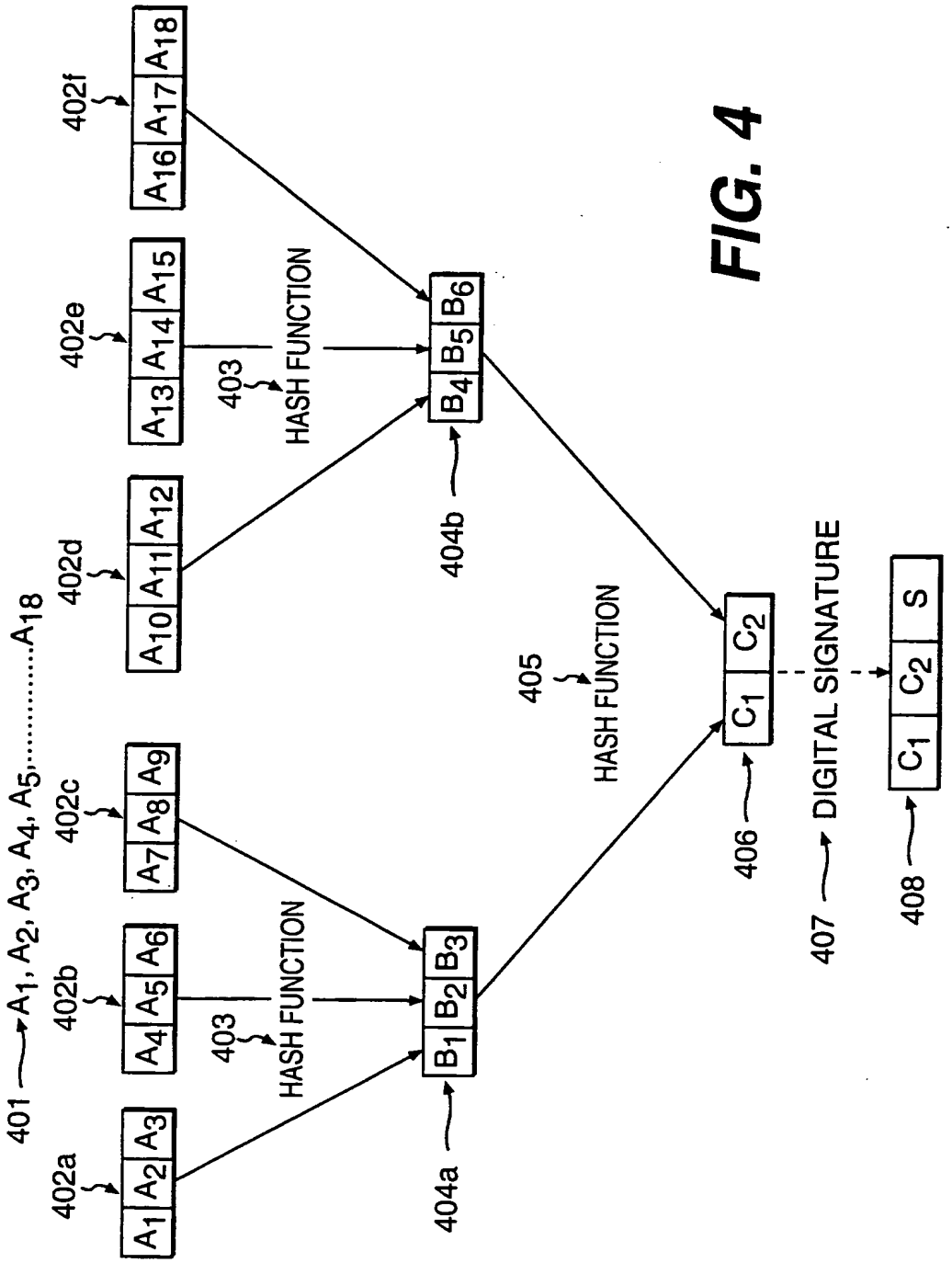

401
$A_1, A_2, A_3, A_4, A_5, \ldots\ldots A_{18}$
402a
A_1 A_2 A_3
402b
A_4 A_5 A_6
402c
A_7 A_8 A_9
402d
A_{10} A_{11} A_{12}
402e
A_{13} A_{14} A_{15}
402f
A_{16} A_{17} A_{18}
403
HASH FUNCTION
403
HASH FUNCTION
404a
B_1 B_2 B_3
404b
B_4 B_5 B_6
405
HASH FUNCTION
406
C_1 C_2
407
DIGITAL SIGNATURE
408
C_1 C_2 S

FIG. 4

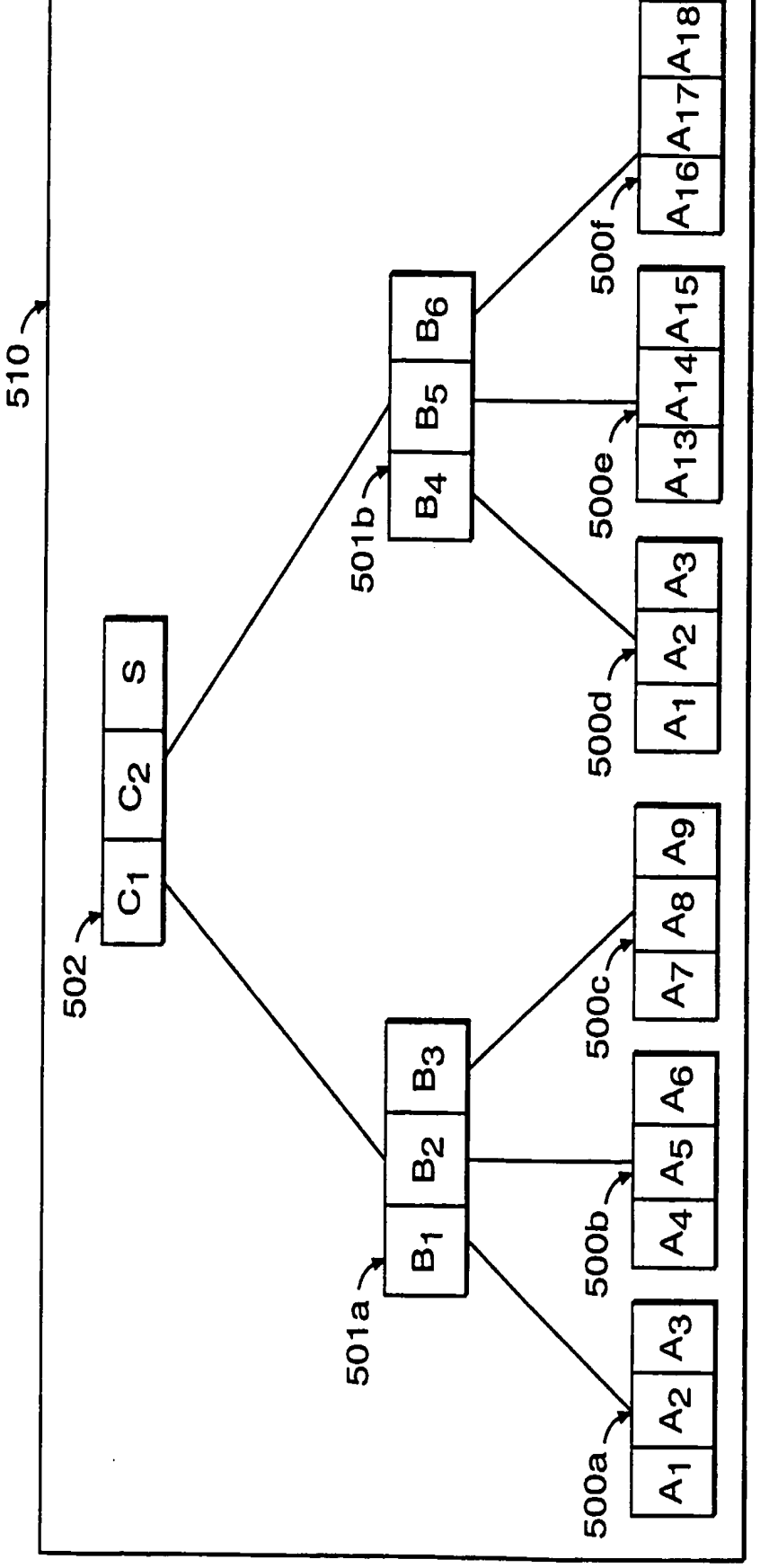

510
502
C1
C2
S
501a
B1
B2
B3
501b
B4
B5
B6
500a
A1
A2
A3
500b
A4
A5
A6
500c
A7
A8
A9
500d
A1
A2
A3
500e
A13
A14
A15
500f
A16
A17
A18

FIG. 5